

Webpack 2: JavaScript 构建工具

作者：王琦

创建日期：2017 年 5 月 16 日

最后修改：2017 年 5 月 16 日

摘要：本文先解释了编写 JavaScript 代码过程中存在的模块依赖问题，再提出了用以自动化决议模块依赖性问题的 Webpack 2 工具。

1. 代码依赖性问题

在 `dependency_problem` 项目中，有三份 JS 代码，它们的依赖关系如下：

```
a.js <--> b.js
  ↑
c.js
```

其中 `a.js` 和 `b.js` 间存在相互依赖关系：

- `a.js` 调用了 `b.js` 的函数 `b()`
- `b.js` 调用了 `a.js` 的函数 `a()`

同时 `c.js` 依赖 `a.js`：`c.js` 调用了 `a.js` 的函数 `a()`。为了在浏览器中运行这些代码，我们有如下 HTML：

```
<html>
  <body>
    <script src="c.js"></script>
    <script src="a.js"></script>
    <script src="b.js"></script>
  </body>
</html>
```

双击打开 `dependency_problem` 目录中的 `index.html`，我们在浏览器的控制台中将发现如下报错信息：

```
Uncaught ReferenceError: a is not defined      -- (1)
    at c.js:1
Uncaught ReferenceError: b is not defined      -- (2)
    at a.js:5
```

错误 (1) 的原因是在于 `c.js` 被引入时，`a.js` 尚未被引入，导致 `c.js` 调用的函数 `a()` 是未定义

的；

错误 (2) 的原因是在于 `a.js` 被引入时，`b.js` 尚未被引入，导致 `a.js` 调用的函数 `b()` 是未定义的。

是的，你可以通过调整 `c.js` 和 `a.js` 的引入顺序，解决问题 (1)。但相信我，无论你怎么调整代码的引入顺序，都将触发问题 2（或它的镜像问题），原因是在于 `a.js` 和 `b.js` 间存在着循环引用关系。

2. 引入 Webpack 解决模块依赖关系

是否存在一种办法，使你免于考虑哪个 JavaScript 文件应该先被 HTML 引入？Webpack 就是为此而设计的：)

Webpack 官方提供了教程：[Getting Started](#)。更简单的，我们在本文也将简述如何使用它。

2.1 安装

在 `webpack` 目录下运行

```
cnpm install --save-dev webpack
```

2.2 配置

参考官方文档 [Getting Started](#) 和 [Configuration](#)，在 `webpack` 目录下创建 `webpack.config.js`：

```

var path = require('path');

module.exports = {
  // Here the application starts executing and webpack starts bundling.
  // Note that this main entry cannot be imported by other modules.
  entry: './app/index.js',

  // options for resolving module requests
  resolve: {
    // directories where to look for modules
    modules: [
      'node_modules',
      path.resolve(__dirname, 'app')
    ]
  },

  // options related to how webpack emits results
  output: {
    // the target directory for all output files
    // must be an absolute path (use the Node.js path module)
    path: path.resolve(__dirname, 'dist'),
    // filename of the output
    filename: 'bundle.js'
  }
};

```

该配置文件定义了：

- Webpack 程序解析的主入口 `./app/index.js`
- 模块的搜索路径： `node_modules` 目录及 `app` 目录。
 - 观察 `c.js` 有 `import {a} from 'a.js';`，亦即当从 `c.js` 引用 `a.js` 的 `a()` 函数时，Webpack 将从 `node_modules` 及 `app` 目录下寻找 `a.js` 文件。
- 输出路径： `dist/bundle.js`

2.3 构建 (Build) 与使用

在 `webpack` 目录下运行

```
webpack
```

观察命令行的输出；同时，你将在 `webpack\dist` 目录下看到新生成的文件 `bundle.js`。你可以在 `index.html` 中引用它：

```
<html>
  <body>
    <script src="dist/bundle.js"></script>
  </body>
</html>
```

双击 `index.html`（用浏览器打开这个页面），查看浏览器的命令行，你将发现——即使你没有在 `index.html` 中显示定义 JavaScript 文件的引入顺序，程序仍然能正确运行——这是由于 Webpack 为你解决了模块间的依赖关系。